PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-243035
(43)Date of publication of application : 02.09.1994

(51)Int.Cl. G06F 12/08
G06F 13/00
G06F 15/16

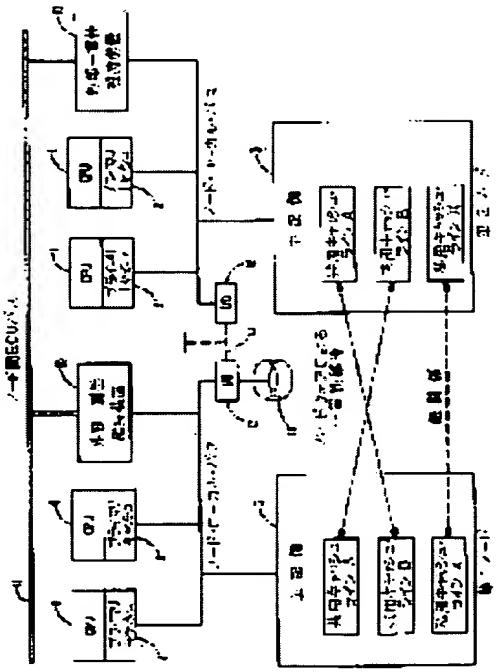
(21)Application number : 05-324668
(22)Date of filing : 22.12.1993

(71)Applicant : BULL HN INF SYST INC
(72)Inventor : HUNTER JOHN C
WERTZ JOHN A

(30)Priority
Priority number : 92 993884 Priority date : 23.12.1992 Priority country : US

(54) GENERALIZED SHARED STORAGE IN CLUSTER ARCHITECTURE FOR COMPUTER SYSTEM

(57)Abstract:
PURPOSE: To improve the performance and generalization of the cluster architecture of a computer.
CONSTITUTION: A 'generalized shared storage' is introduced to a cluster architecture, and this generalized shared storage is held in a state with coherency by a coherency maintaining mechanism constituted of hardware which adds an operation to a shared object when the positioning or reference or the like of the shared object is operated. Each of plural nodes is provided with a central processing unit 1 having a primary cache 2, local main storage 3, outside coherency maintaining device (ECU) 10, and node local bus 4 connecting them. Those nodes are connected through an inter-node ECU bus 11. The ECU monitors the bus, detects a cache line request, and executes a coherency maintaining operation according to the state of the requested cache line. The rate of a private storage/shared storage can be arbitrarily and dynamically changed in each node in the cluster.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than

the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

*1

(19) 日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11) 特許出願公開番号

特開平6－243035

(43)公開日 平成6年（1994）9月2日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
G06F 12/08		H 7608-5B		
13/00	355	7368-5B		
15/16	310	M 7429-5L		

審査請求 未請求 請求項の数9（全 14 頁）

(21)出願番号　特願平5－324668

(22)出願日　平成5年（1993）12月22日

(31)優先権主張番号　993884

(32)優先日　1992年12月23日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　391031270
ブル・エイチエヌ・インフォメーション・システムズ・インコーポレーテッド
BULL HN INFORMATION SYSTEMS, INCORPORATED
アメリカ合衆国マサチューセッツ州 01821, ビレリカ, テクノロジー・パーク （番地なし）

(72)発明者　ジョン・シー・ハンター
アメリカ合衆国アリゾナ州85021, フェニックス, ウエスト・エコー・レーン 121

(74)代理人　弁理士 湯浅 恭三 （外5名）

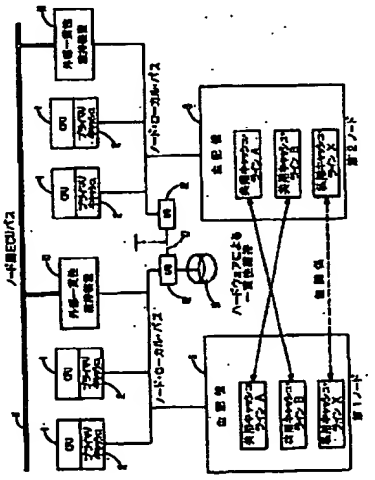
最終頁に続く

(54)【発明の名称】コンピュータ・システムのためのクラスタ・アーキテクチャにおける一般化共用記憶

(57)【要約】

【目的】　コンピュータのクラスタ・アーキテクチャの性能及び汎用性を向上させる。

【構成】　クラスタ・アーキテクチャに「一般化共用記憶」を導入し、この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェクトに対して操作を加えるハードウエアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。複数のノードの各々が、プライマリ・キャッシュ２を有する中央処理装置１と、ローカル主記憶３と、外部一貫性維持装置（ＥＣＵ）１０と、それらを結合するノード・ローカル・バス４とを備えている。ノードどうしはノード間ＥＣＵバス１１によって接続される。ＥＣＵはバスを監視してキャッシュ・ライン要求を検出し、要求されたキャッシュラインの状態に応じて一貫性維持動作を実行する。私用記憶／共用記憶の割合をクラスタ内の個別のノードにおいて任意に且つ動的に変化させることができる。



1

【特許請求の範囲】
【請求項1】　コンピュータのクラスタ・アーキテクチャにおいて、
A）複数のノードと、
B）前記複数のノードの各々に少なくとも1個が装備され、その各々が一貫性維持特性を有している、中央処理装置と、
C）前記複数のノードの各々に装備され、その各々が、みずからが装備されているのと同じノードに装備されている全てのプライマリ・キャッシュを結合している、ノード・ローカル通信手段と、
D）前記複数のノードの各々に装備され、その各々が、共用物理空間と非共用物理空間とに割当て可能な物理空間を有している、ローカル主記憶と、
E）前記複数のノードの各々に装備され、その各々が、みずからが装備されているのと同じノードに装備されている前記ノード・ローカル通信手段及び前記ローカル主記憶と結合している、外部一貫性維持装置と、
F）全ての前記外部一貫性維持装置を結合しているノード間通信手段と、を備えており、
G）前記外部一貫性維持装置の各々が、
1）当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ノード間通信手段と前記ノード・ローカル通信手段との両方を監視するための、監視手段と、
2）一貫性維持手段であって、
a）前記監視手段が、当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ノード・ローカル通信手段上に送出されたキャッシュ・ライン要求を検出し、且つ、検出したそのキャッシュ・ライン要求が非共用キャッシュ・ライン要求であると判定したときに、前記監視手段がそのように判定したことに応答して、その非共用キャッシュ・ライン要求を、前記ローカル主記憶の前記非共用物理空間へ導き、一方、
b）前記監視手段が、当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ノード・ローカル通信手段上に送出されたキャッシュ・ライン要求を検出し、且つ、検出したそのキャッシュ・ライン要求が共用キャッシュ・ライン要求であると判定したときに、前記監視手段がそのように判定したことに応答して、その共用キャッシュ・ライン要求に関するキャッシュ・ラインの一貫性状態を調べることにより、その共用キャッシュ・ライン要求に応えるためにはノード間動作が必要か否かを判定し、そして、そのノード間動作が必要であると判定されたならば、前記ノード間通信手段を介して、一義的識別子と一貫性維持コマンドとを、その他全ての前記外部一貫性維持装置へ送出するようにした、
前記一貫性維持手段と、
を備えており、

2

以上によって、クラスタ内の全ての前記ローカル主記憶の前記共用物理空間内で得られる最新にインスタンス化された要求データが、前記外部一貫性維持装置のうちのそのデータを要求した外部一貫性維持装置へ提供されるようにした、ことを特徴とするコンピュータのクラスタ・アーキテクチャ。
【請求項2】　前記外部一貫性維持装置の各々が、前記クラスタの共用記憶空間内に格納されている情報の各単位ごとに一貫性状態を付与するようにしてあり、付与する一貫性状態が、
A）当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ローカル主記憶の前記共用記憶空間内に存在している要求された情報のコピーが、前記クラスタ内に存在している唯一のコピーであることを示す「専有」状態と、
B）当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ローカル主記憶の中に存在している要求された情報のコピーが、その同じノードに装備されている中央処理装置によって変更されていることを示す「変更済」状態と、
C）当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ローカル主記憶の中に存在している要求された情報のコピーが、不在であるか、または、最新のものとは異なっていることが知られていることを示す「無効」状態と、
D）当該外部一貫性維持装置が装備されているのと同じノードに装備されている前記ローカル主記憶の中に存在している要求された情報のコピーが、複数のノードの中に存在している要求された情報の複数の現在コピーのうちの1つであることを示す「共用」状態と、を含んでいることを特徴とする請求項1記載のコンピュータのクラスタ・アーキテクチャ。
【請求項3】　前記外部一貫性維持装置の各々に備えられている前記一貫性維持手段の各々が、共用情報の1つのブロックに関連したローカル・キャッシュ・ライン要求に応答して、
A）要求されたキャッシュ・ラインの状態が「専有」状態または「変更済」状態であったならば、要求されたブロックを要求元ノードに装備されている前記ローカル主記憶から直接取り出すようにし、この場合、要求元ノードに装備されている前記ローカル主記憶の中以外には、前記クラスタ内のどこにも現在コピーは存在していないため、いかなる遠隔動作も行なわず、
B）要求されたキャッシュ・ラインの状態が「無効」状態であったならば「書込みを前提とした読取り」コマンドを前記通信手段を介してその他の全てのノードへ送出するようにし、この場合、要求されたキャッシュ・ラインの情報を「専有」状態または「変更済」状態で保持している遠隔ノードがそれに応答して、要求されたキャッシュ・ラインの情報を前記通信手段を介して要求元ノー

ドへ移転し、
C）要求されたキャッシュ・ラインの状態が「共用」状態であったならば「無効化」コマンドをその他の全てのノードへ送出することによって、その他の全てのノード内において、要求されたキャッシュ・ラインの状態を「無効」状態にセットさせる、ようにしてあることを特徴とする請求項２記載のコンピュータのクラスタ・アーキテクチャ。
【請求項４】　前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項１記載のコンピュータのクラスタ・アーキテクチャ。
【請求項５】　前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項２記載のコンピュータのクラスタ・アーキテクチャ。
【請求項６】　前記ノード間通信手段が、ノード間外部一貫性維持装置バスを含んでいることを特徴とする請求項３記載のコンピュータのクラスタ・アーキテクチャ。
【請求項７】　前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項１記載のコンピュータのクラスタ・アーキテクチャ。
【請求項８】　前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項２記載のコンピュータのクラスタ・アーキテクチャ。
【請求項９】　前記ノード間通信手段が、直接結合手段を含んでいることを特徴とする請求項３記載のコンピュータのクラスタ・アーキテクチャ。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はデータ処理技術に関し、より詳しくは、クラスタ・アーキテクチャに用いる一般化共用記憶に関する。
【０００２】
【従来の技術】一般的に、クラスタ・アーキテクチャと呼ばれているコンピュータ・アーキテクチャでは、複数のノードの各々に、少なくとも１つの（そして通例は複数の）中央処理装置（ＣＰＵ）と、ローカル記憶と、入出力装置（Ｉ／Ｏ）と、その他の要素とを装備している。良好な評価を受けているクラスタ・アーキテクチャのうちの１つに、物理記憶アドレス空間を恒久的に２つの領域に分割しておくようにしたクラスタ・アーキテクチャがあり、そこでは、分割した下半分の領域（最上位ビット＝「０」の領域）を、当該ノードに装備されているプロセッサだけがアクセスすることのできるローカル記憶とし、一方、分割した上半分の領域（最上位ビット＝「１」の領域）は共用記憶を画成するようにしており、この共用記憶は物理的に集中化してあって、全てのノードがそれにアクセスすることができる。ある共用変数の複数のコピーが別々のノードに保持されているときに、それら複数のコピーの間の一貫性を維持するためのハードウェア機構は装備されていない。そのため一貫性の維持は完全にソフトウエアに任されている。
【０００３】以上のアーキテクチャを拡張したものが、「Bull NH 共用バッファ・アーキテクチャ」（以後、共用バッファ・アーキテクチャを「ＳＢＡ」と略称する）であり、その拡張とは、物理的に集中化してある共用記憶に本来所属するデータの、ただしクラスタ内の別々のノードのキャッシュ記憶の中に保持されている、複数のコピーの間の一貫性（コヒーレンシー）を維持するためのハードウエア機構を導入したことにある。これによって、別々のノードが、共用データに対するある種の操作を互いに並行的に、より短いアクセス時間で実行できるようになっており、それによってアーキテクチャの性能が改善されている。
【０００４】ＳＢＡの変形構成の１つである「Bull NH 分散共用バッファ・アーキテクチャ」（以下、分散共用バッファ・アーキテクチャを「ＤＳＢＡ」と略称する）は、集中化した共用記憶に代えて分散した共用記憶を用いており、この構成形態は、性能及び可用度、それに既存のある種のコンピュータ製品との間の互換性という点に関して、多くの利点を有している。私用記憶と共用記憶との間のアドレス区分は固定されていて不変であるが、共用記憶を各コンピュータ・ノード内に複製してあるため、全てのノードが同時に読取りアクセスを行なうことができる。またハードウエアで構成した一貫性維持機構によって、分散した夫々の記憶の中のデータどうしの間の一貫性を維持するようにしている。
【０００５】
【発明が解決しようとする課題】公知のアーキテクチャの変形構成である上述のアーキテクチャは、それらのいずれもが、私用記憶と共用記憶との間の区分を固定して不変にしていた。この特性のために、クラスタの個々のノードの内部でも、また複数のノードの間でも、私用記憶及び共用記憶を構成するに際して、そのときどきアプリケーション・ミックス（アプリケーションの組合せ）の要求に適合するように、それら２通りの記憶を異なった大きさに構成することが（たとえ全く不可能でないにしても）困難であった。共用アプリケーションを実行するのに私用記憶を使用することはできず、一方、私用アプリケーションを実行するのに共用記憶を使用したならば、その私用アプリケーションは不必要なコヒーレンシー・トラフィックに煩わされる上に、完全性に関する問題を生じるおそれもある。本発明はこれらの制約を克服するものである。
【０００６】従って本発明の広い目的は、改良したクラスタ・アーキテクチャを提供することにある。
【０００７】本発明のより具体的な目的の１つは、任意の大きさとすることができクラスタ内の任意の個数のノードによって共用させることのできる不連続な共用オブジェクトを保持する一般化共用記憶を用いるようにしたクラスタ・アーキテクチャを提供することにある。

【０００８】本発明の更に別の具体的な目的は、クラスタ内の全ての要素が共用記憶を極めて迅速に入手できるようなクラスタ・アーキテクチャを提供することにある。

【０００９】

【課題を解決するための手段】要約して述べるならば、本発明の以上の目的並びにその他の目的は、クラスタ・アーキテクチャに「一般化共用記憶」を導入することによって達成される。この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェクトに対して操作を加えるハードウエアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。これによって、私用記憶／共用記憶の割合をクラスタ内の個別のコンピュータ・ノードにおいて任意に且つ動的に変化させることができることから、アーキテクチャの性能と汎用性との両方が向上する。

【００１０】

【実施例】先ず最初に、従来例に係るクラスタ・アーキテクチャを示した、図１のハイレベルのブロック図について説明する。図１に例示したシステムは、１６個の中央処理装置（ＣＰＵ）１を、各ノードが４個ずつのＣＰＵを備えた４つのノードに編成した構成を有する。各ＣＰＵは、その他のシステム構成要素との間の通信に関して一貫性を維持する特性を有しており、また通常そうであるように、プライマリ・キャッシュ２を装備している。更に、各ノードはローカル共用記憶３を含んでおり、ローカル共用記憶３は、そのノード内の夫々のＣＰＵ１との間でプライマリ・キャッシュを介して通信を行なう際に、ノード・ローカルな通信システムを介してその通信を行ない、図示例におけるノード・ローカルな通信システムは、ノード・ローカル・バス４である。加えて、各々のノード・ローカル・バス４は、クラスタ共用記憶５にも結合している。ローカル共用記憶３の各々とクラスタ共用記憶５とは同一の大きさにしても良く、例えばそれらを同じ「２ギガバイト」とすることができる。記憶をアドレスするためのアドレス構造は、次のようなものであり、即ち、あるＣＰＵが発した記憶アドレスの最上位ビット（ＭＳＢ）が「０」であったならば、その記憶アドレスは、そのＣＰＵが所属しているノードのローカル共用記憶の中に格納されている情報をアドレスするためのものであり、また、その記憶アドレスのＭＳＢが「１」であったならば、その記憶アドレスは、クラスタ共用記憶の中に格納されている情報をアドレスするものであると定めてある。ノード間通信はＩ／Ｏ装置１２及びノード間バス１７を介して行なわれ、通信をするノードどうしの間の距離がかなりのものであるならば、通信バス２０を用いれば良く、以上のことは全て、当業界においては公知の事柄である。

【００１１】以上に説明したアーキテクチャは、適当な操作を行なうことによって、クラスタ共用記憶５を介してクラスタ内の全域において情報交換を行なえるという利点を有する。しかしながらこのアーキテクチャには、その利点と同時に、幾つかの短所も付随している。それら短所は例えば次のようなものである。

A）私用記憶と共用記憶との区分が恒久的に固定されており、そのため、実行すべきアプリケーション・ミックスが、私用記憶を用いた方が良好に機能するアプリケーションと、共用記憶を用いた方が良好に機能するアプリケーションとの両方を含んでいる場合には、非常な非効率を生じることがある。

B）クラスタ共用記憶には、シングル・ポイント・オブ・フェイリャが存在している。

C）各々のノードごとに、そのノードのためのオペレーティング・システムを装備しておく必要がある。

D）クラスタ共用記憶への個々のアクセスの各々ごとに、全てのノードの記憶管理機構の同意を得る必要がある。

【００１２】既述の如く、図１に示したアーキテクチャに変更を加えてシステム性能の漸増的改善を達成した変形構成態様が幾つか存在している。それらのうち、共用バッファ・アーキテクチャ（ＳＢＡ）に該当する構成態様としては、集中化した共用記憶に本来所属するデータの、ただしクラスタ内の別々のノードの夫々のキャッシュ記憶の中に保持されている、複数のコピーどうしの間の一貫性を維持するハードウエア機構を装備したものがある。これによって、別々のノードが、共用データに対する複数の操作を互いに並行的に、より短いアクセス時間で実行できるようになっているため、アーキテクチャの性能が向上している。また、分散共用バッファ・アーキテクチャ（ＤＳＢＡ）としては、集中化した共用記憶に代えて分散共用記憶を用いたものがあり、この構成形態は、性能及び可用度、それに既存のコンピュータ製品との間の互換性という点に関して、多くの利点を有している。私用記憶と共用記憶との間のアドレス区分は固定されていて不変であるが、共用記憶を各コンピュータ・ノード内に複製してあるため、全てのノードが同時に読取りアクセスを行なうことができる。またハードウエアで構成した一貫性維持機構によって、分散した夫々の記憶の中のデータどうしの間の一貫性を維持するようにしている。

【００１３】これらに対して、本発明は、クラスタ・アーキテクチャに、根本的な新機軸を提供するものである。本発明を理解するには、その前に先ず、本発明の重要な一局面である「共用オブジェクト」の概念を明確に認識しておく必要がある。共用オブジェクトは、任意の大きさとすることができ、また、クラスタ内の任意の個数のノードの間で共用させることができる。複数の共用オブジェクトの集合体によって「一般化共用記憶」が構成され、この一般化共用記憶は、共用オブジェクトの位置付けや参照等が行なわれたときにその共用オブジェク

トに対して操作を加えるハードウェアで構成した一貫性維持機構によって、一貫性を有する状態に維持される。

【００１４】共用オブジェクトは、共用し得るという性質を備えたオブジェクトであると定義することができる。共用オブジェクトは、クラスタ内の全てのノードに知られている。共用オブジェクトのエクステント（存続範囲）と、アクセスを共用することを許可されているプロセスとは、記述子によって明示される。共用オブジェクトの実現形態は、目的オペレーティング・システムの種類によって様々に異なり、ファイルの形態とされることもあれば（例えば「Bull NH 社」の「GCOS 8」オペレーティング・システム等の場合）、ストリームの形態とされることもある（例えば「UNIX」オペレーティング・システム等の場合）。共用オブジェクトは各ノードごとに個別に、そのノードの仮想空間内でインスタンス化され、それによって、異なったノードの夫々のプロセスによって共用され得るようになる。仮想空間内でのインスタンス化が行なわれたならば、共用オブジェクトは（例えばコマンドによって）、分散共用記憶を用いたクラスタの別々のノードの夫々の記憶の中で、各ノードごとに個別に、物理的にインスタンス化することができるようになる。集中共用クラスタ記憶の中では、ある１つの共用オブジェクトの幾つもの物理的なインスタンス化が行なわれることもあり、例えば、複数の「UNIX」プロセスの間で共用されるページをサポートする共用オブジェクト等がそうである。共用オブジェクトは以下のものを備えている。

A）任意長表示の一義的名標
B）固定長表示の一義的識別子（ＵＵＩＤ）
C）アドレス可能な内部要素
D）参照許可

【００１５】ＤＳＢＡ環境に即して考察することが、共用オブジェクトに対するリアルタイムの操作を理解するための最も容易な方法である。図２を参照して説明すると、クラスタ内の各ノード（図を見易くするために、図２にはノードのうちの２つだけ示してあり、また各ノード内のＣＰＵ１のうちの２個だけを示してある）は、外部一貫性維持装置（External Coherency Unit:ＥＣＵ）１０を装備している。ＥＣＵ１０は（ａ）そのノードのノード・ローカル・バス４を監視しており、そのノード・ローカル・バス４上にその他のノードの中にも存在している「共用」キャッシュ・ラインに関するコマンドが送出されたときにそれを検出し、そして（ｂ）それに該当するコマンドを検出したならば、条件に従ってそのコマンドを（そのキャッシュ・ラインを表わす一義的識別子を使用して）その他のノードへ例えば独立したノード間ＥＣＵバス１１等を介して転送する。（更にはポイント・ツウ・ポイント接続も可能であり、そのためには、各々のＥＣＵの中のディレクトリを用いてどのノードがコピーを保持しているかを追跡するようにする）。一方、その他の全てのＥＣＵが（ａ）ノード間ＥＣＵバス１１を監視しており、そのノード間ＥＣＵバス１１上に、それら夫々のＥＣＵのノードのローカル記憶の中に存在しているキャッシュ・ラインに影響を及ぼすコマンドが送出されたときにそれを検出し、また（ｂ）それに該当するコマンドを検出したならば、そのコマンドを、そのノードにおける物理タグへ変換し、そして（ｃ）変換したコマンドを、そのノードにおけるノード・ローカル・バス上へ送出する。概念説明のための図示の具体例では、ＥＣＵ１０を一般的な一貫性維持装置を用いて構成することも可能であり、その装置において、一義的識別子を有するキャッシュ・タグ・ディレクトリと、そのノードで現在インスタンス化されている各「共用」キャッシュ・ラインの状態を表わす一貫性状態（例えば、変更済、専有、共用、無効）とを用いるようにすれば良い。

【００１６】具体的な動作例について説明すると、例えばクラスタ内の、ある１つのノードの中のある１つのＣＰＵが、ある１本の共用キャッシュ・ラインの内容を変更しようとする場合、そのＣＰＵは、何よりも先ず最初に、専有所有権を獲得する必要がある。そのためのコマンドがローカルＥＣＵ１０によって監視されており、もし、そのノードが専有所有権を持っていなかったならば、そのコマンドはその他の全てのノードのＥＣＵへ転送される。すると、それらノードのうち、変更されていない有効なコピーを保持していた各ノードは、（例えばそのノードのプライマリ・キャッシュ２、及び／または、ローカル・メイン記憶３の中にそれまで保持していた）全てのローカル・コピーを無効化する。もし、あるノードがそのキャッシュ・ラインを「変更済」状態で保持していたならば、そのノードは、その「変更済」状態のキャッシュ・ラインを要求元ノードへ移転した上、その移転元ノードの中でそのキャッシュ・ラインを無効化する。これによって、その（移転された）キャッシュ・ラインが専有的に所有されて、更新可能な状態になる。以上の一連の一貫性維持作業により、共用オブジェクトを参照しているプロセスは、それがどのノードにおいて実行されているどのプロセスであっても、最新のデータを受け取ることができるように保証されている。

【００１７】以上のアーキテクチャによれば、Ｉ／Ｏ装置１２は、ある共用オブジェクトの全て或いは一部を保持することを割当てられているローカル物理記憶との間で適正に作業を行なうことができる。即ち、Ｉ／Ｏ装置１２が共用記憶からの読込みを行なうときには、（そのＩ／Ｏ装置１２が装備されているノード以外の）その他のノードの共用記憶の中にあった最新データが自動的に移転される。また、Ｉ／Ｏ装置１２が共用記憶への書込みを行なうときには、そのＩ／Ｏ装置１２が装備されているノードにおける専有所有権が、データの到着と共に、自動的に獲得され、そして、Ｉ／Ｏ装置１２がその

作業を完了した後には、いずれかのノードのプロセッサがそのデータを参照したならば、そのデータが自動的にそのノードの記憶へ移転される。この一般化共用記憶を採用することによって得られる利点は次のとおりである。・ノードの内部でもノードどうしの間でも物理記憶をより効率的に使用し得ることに関する利点：

（１）物理的インスタンス化した私用記憶領域及び共用記憶領域が非連続領域であっても良く、それら領域の合計が物理記憶の大きさを超えない限り、それら領域を任意の大きさにすることができる。

（２）同一時刻における夫々のノードの私用／共用の割合を互いに異ならせることができる。

（３）記憶の合計の大きさを、単一のオペレーティング・システムないしプラットフォームの物理的アドレス能力以上にすることができる。

・クラスタのソフトウエアが簡明化されることに関する利点：

（１）共用記憶を各ノードごとに個別に管理するため、大域的記憶管理機構が不要であり、また、共用領域を管理するために複数のノードが互いに協働して作業する必要もない。

・可用度が向上することに関する利点：

（１）ＳＢＡの集中共用記憶に存在しているシングル・ポイント・オブ・フェイリャが、ここには存在していない。

（２）あるノードが故障した場合に、そのノードの私用記憶を強制的に共用可能にすることにより、別のノードがその私用記憶領域にアクセスして故障からの回復を図ることができる。

・コヒーレンシー・トラフィックが軽減されることに関する利点：

（１）ノード間コヒーレンシー・トラフィックが発生するのは、非専有的に所有されている共用オブジェクトへの書込みが行なわれるときと、存在していない共用オブジェクトが参照されるときとの、いずれかの場合に限られる。

（２）共用オブジェクトのコピーを保持していないノードからは、コヒーレンシー・トラフィックが排除されている。

【００１８】なお付言すると、この「分散した」一般化共用記憶の性能は「集中化した」共用記憶の性能を凌駕し得るものであり、なぜならここに提案した機構は、共用データがキャッシュ方式の高速性という利点を享受し得るようにすると同時に、伝統的なアドレス可能な記憶をそのまま残しているからである。即ち、この機構によれば、個別の複数のノードが、同じ共用変数の、ただしそれら夫々のノードがみずからの中に保持しているところの、複数のコピーを、各々のローカル・バスが発揮し得る限りの高速でもって互いに同時に読み取ることができるのである。これに対して、「Ａｂｒｏｊｏ」やＳＢＡに用いられている集中共用記憶は、同時に多数の読取り要求が発せられた場合には、みずからがボトルネックとなってしまう。

【００１９】「共用」し得るものか否かを検出するための基本的な方式として２通りの方式があり、その１つは「ハードウエア方式」、もう１つは「ソフトウエア方式」である。

ハードウエア方式：ページ・テーブル記述語（Page Table Descriptor Word:ＰＴＷ）に、あるページが共用ページであるか否かを表示するための１つのビット（「共用」ビット）を含ませておく。共用領域の参照が行なわれるときにはそのＰＴＷがＣＰＵの中に存在しており、このＣＰＵがノード・ローカル・バス上へ「共用」ビットを送出することによって、そのノードに装備されているＥＣＵに、共用領域の参照が行なわれることを通知する。こうすることによって、ＥＣＵを、共用ページの物理タグに関するキャッシュとして動作させることができ、このＥＣＵキャッシュの中には、高頻度でアクセスされるタグだけを実際に保持しておくようにする。参照頻度の低いタグはＥＣＵが記憶から取り出すようにしておく。

この方式の長所：簡明、高速、低コスト、そしてソフトウェア独立（即ち、ソフトウエアに左右されることがない）である。

この方式の短所：ＣＰＵ及びバスのハードウエアに改造を施す必要がある。

【００２０】ソフトウエア方式：現在インスタンス化されている共用ページのＰＴＷ（実際にはＥＣＵのエントリ）の全てをＥＣＵ内のバッファ記憶の中に保持しておき、このＥＣＵが、監視を行なうことによって（或いは、ＥＣＵどうしを直接接続してディレクトリを参照することによって）共用キャッシュ・ラインを検出するようにしておく。共用ページのインスタンス化または削除が行なわれる度に、記憶管理ソフトウエアが、ＥＣＵ内のＰＴＷを保持しているバッファに対して更新を施すようにする。

この方式の長所：中心的なシステム・ハードウエアに改造を施す必要がない。ＰＴＷに余分なビットを付加する必要がない。そして「既存のハードウエアにも装備し得る可能性を有する」。

この方式の短所：記憶管理ソフトウエアに改造を施す必要があり、ハードウエアのコストが増大する。

【００２１】これより、図３の、より詳細な具体例のブロック図について説明して行く。ＥＣＵ１０は連想ディレクトリ１３を含んでおり、この連想ディレクトリ１３の機能はノード・ローカル・バス４上のアドレス並びにノード間ＥＣＵバス１１上のアドレスを常時監視することにある。ここでは、このディレクトリ１３の中に、同じＥＣＵの中のローカル記憶３内に現在物理的にインスタンス化されている共用ページのの各々に１つずつが対

応した、ＥＣＵエントリが保持されているものとする。ここで図６を参照して簡単に説明しておくと、ＥＣＵエントリは、それが対応しているページのローカル物理アドレスを包含していると共に、そのページの中に含まれている複数本のキャッシュ・ライン（例えば、４０９６バイトの大きさのページの中には連続した６４バイトから成るキャッシュ・ラインが６４本存在している）の各々に対応した一義的識別子及び一貫性状態ビット（２～３ビット）のビット集合を包含している。１つのＥＣＵエントリの全体の大きさは２４バイトから３２バイトまでの間の大きさであり、これが１つの共用ページに対応する。

【００２２】説明を再び図３に戻し、ＥＣＵ１０は、非共用のキャッシュ・ラインに関する要求のコマンドは無視する。そのためその種のコマンドは、そのまま、ローカル記憶３の非共用物理空間１４の中の指定された記憶をアドレスすることになる。一方、ＥＣＵ１０が識別したコマンドが、共用キャッシュ・ラインに関するコマンドであった場合には、ＥＣＵ１０は、ＥＣＵディレクトリ１６の中に格納されているその共用キャッシュ・ラインの一貫性状態（「変更済」、「専有」、「共用」、または「無効」）を調べることによって、一貫性を維持するためには何らかのノード間動作を行なうことが必要か否かを判定する。そして、ノード間動作が必要であると判定されたならば、物理アドレスを然るべき一義的識別子に変換した上で、その変換した識別子を、該当するコマンドと共に、ＥＣＵバスを介してクラスタ内の他のノードへ転送する。

【００２３】一例として、あるＣＰＵが、ある共用キャッシュ・ラインの専有コピーを獲得しようとする場合、そのＣＰＵは、然るべきコマンド（例えばＲＴＷコマンド等である。ＲＴＷコマンドとは「書込みを前提とした読取り」コマンドの意味であり、これについては具体例のシステムに関連して後に更に詳細に説明する）を、そのＣＰＵが所属しているノードのノード・ローカル・バス上に送出する。すると、そのノードのＥＣＵ１０が以下の動作のうちのいずれかを実行する。

（１）そのキャッシュ・ラインの状態が「専有」状態または「変更済」状態であったならば、そのキャッシュ・ラインをローカル記憶３から直接読み出して、要求元プロセッサへ転送する。この場合、そのキャッシュ・ラインのコピーは他のどこにも存在していないため、遠隔動作は何も実行しない。

（２）そのキャッシュ・ラインの状態が「無効」状態であったならば、ノード間ＥＣＵバス１１を介して他のノードへＲＴＷコマンドを送出する。ある遠隔ノードが、そのキャッシュ・ラインを「専有」状態または「変更済」状態で保持していたならば、その遠隔ノードは、ノード間ＥＣＵバスを介してそのキャッシュ・ラインを要求元ノードへ転送してくる（即ち、移転が行なわれる）。また複数のノードが、そのキャッシュ・ラインを「共用」状態で保持していたならば、それら複数のノードの全てがそのキャッシュ・ラインを送出しようとする。しかしながら、ＥＣＵバスの標準的な優先権ロジックが、それら送出のうちの１つを選択してその他をキャンセルする。更に、全ての遠隔コピー（遠隔ノードに保持されているコピー）が「無効」状態にセットされる。

（３）そのキャッシュ・ラインの状態が「共用」状態であったならば、ＩＮＶコマンドを他のノードへ送出することによって、他のノードが夫々みずからの中に保持しているそのキャッシュ・ラインのコピーを「無効」状態にセットするようにする（この場合には、要求元ノードの中に既に現在コピーが存在しているため、移転が行なわれる必要はない）。

【００２４】バスからの入力とキャッシュ・ラインの状態との組合せが以上のものとは異なる様々なものである場合に、夫々に一貫性を維持するために必要となるその他のノード間動作も存在しているが、それらについては、後に要約して述べることにする。一貫性維持のための以上の処理手順は、ハードウェア・レベルで動作するものであり、そして、共用ページを参照するプロセスが、それがどのノードにおいて実行中のどのプロセスであっても、その共用ページの全体に亙って最新のデータを見られるようにすることを保証するものである。

【００２５】ページ・イン方式：経済性という観点からの要請に従うならば、例えばページ・フォールトを発生させるような参照（即ち、無効ＰＴＷ）がなされたとき等に行なう、共用ページの物理的インスタンス化は、実際にその共用ページを参照するノードの中だけで行なわれることが望ましい。これに関して従来のシステムでは、記憶管理機構ないし記憶管理ソフトウェアが、物理ページ及びＰＴＷのインスタンス化を行なった後に、Ｉ／Ｏ装置に、例えば従来のＩ／Ｏ通信チャネル１７等を介してそのページの内容をページ・インさせるようにしていた。ところが、分散記憶システムの場合には、そのページの内容が既に他のノードの中に存在している可能性もある。そこで、記憶管理機構ないし記憶管理ソフトウェアは、通常のページのインスタンス化の作業を行なうことに加えて、そのページの内容がどこかに存在していないかを調べるサーチも行なわせることが必要となる。これらのことを行なうには、（１）ＥＣＵの中に、全てのキャッシュ・ラインに「無効」のマーキングを施したＥＣＵページ・エントリを格納した後に、（２）ページ・フォールトを発生させた記憶位置への再度の参照を（今度は有効ＰＴＷによって）試みるようにする。これを行なったならば、その結果は（１）要求されたキャッシュ・ラインが別のノードからその記憶の中のページ・フレームへ移転されるか、或いは、（２）そのページはどのノードの共用記憶の中にも存在していないと判定されるかの、いずれかとなる。結果が（２）となった場

合には、記憶管理ソフトウェアは、そのページを保持している記憶装置（例えばディスク装置）へのＩ／Ｏアクセス権をどのノードが持っているのかを判定した上で、そのノードがＩ／Ｏ動作を実行してそのディスク装置からそのノードみずからの記憶の中へそのページをページ・インすることを（例えば特別の割込等によって）要求する必要がある。そのＩ／Ｏ動作が完了したならば、本来実行していたソフトウエアを、問題となっているその参照が行なわれた点から再始動させることができるようになり、また夫々のノードのＥＣＵが、移転によってその参照を完了させることができるようになる。

【００２６】ページ・アウト方式：夫々のノードにおけるページ置換の判断は、個々のノードごとに、他のノードからは独立して行なえるようにしておくことが望ましい。複数のノードにおいてインスタンス化されている共用ページの場合には、例えばそのページが極めて低頻度でしか参照されないノードがあるなら、そのノードのローカル記憶からは、そのページを除去できるようにしておくことが望まれる。そこで、ある長さの時間に亙って一度も参照されていない共用ページがあって、その共用ページが変更済のものであるならば、そのページのホーム・ノードへ（例えば特別の割込等によって）、そのページをディスク装置１８（図３）へ書込むべきことを伝えることによって、そのページを除去できるようにしている。このとき、そのホーム・ノードの記憶管理ソフトウェアは、もし、そのページを物理的にインスタンス化したものがホーム・ノードの中に存在していなかったならば、先ずそのページの物理的インスタンス化を行ない、その後にＩ／Ｏ動作を開始して、そのページの中の各々のキャッシュ・ラインを参照して行きながらそのページの最新の完成コピーを他のノードから集めるようにする。遠隔ノードの中のキャッシュ・ラインの一貫性状態は、以前のままに放置しておくようにしても良く、或いは、夫々の遠隔ノード内で「無効」状態にセットするようにしておいても良い。後者の方式とすれば、そのページが低頻度でしか参照されないノードにおいて、そのページを削除することが容易になる。（ＰＴＷが変更済ページであることを示している場合であっても、ＥＣＵに対して照会を発することによってそのページ内の全てのキャッシュ・ラインが無効であることを判定できるようにしてあれば、記憶管理ソフトウェアはそのページを安全に廃棄することができる）。これ以後、そのページを参照しようとするノードが現われたならば、そのノードは、ホーム・ノードの中に存在しているコピーの移転によって、そのページのデータを受け取ることになる。

【００２７】これより、図４及び図５について説明する。これらの図は、ＣＰＵ１と、そのＣＰＵ１のプライマリ・キャッシュ２と、ノード・ローカル・バス４と、ＥＣＵ１０と、ノード間ＥＣＵバス１１との間のインターフェースを示した状態図であり、従って、同機種から成る「GCOS 8」環境に組み込めるようにしたＥＣＵ１０を規定している状態図である。この「GCOS 8」環境で使用される様々な用語の意味は次のとおりである：

・第１レベルのバスとは、ＣＰＵ１と、そのＣＰＵ１のプライマリ・キャッシュ２とを結合しているバスのことである。

・第２レベルのバスとは、ノード・ローカル・バス４のことである。

・第３レベルのバスとは、ノード間ＥＣＵバス１１のことである。

・データ移動コマンドに関しては次のとおり：

Ｉ１－第１レベルのバスから受取ったデータ。

Ｏ１－第１レベルのバスへ出力されたデータ。

Ｉ２－第２レベルのバスから受取ったデータ。

Ｏ２－第２レベルのバスへ出力されたデータ。

Ｉ３－第３レベルのバスから受取ったデータ。

Ｏ３－第３レベルのバスへ出力されたデータ。

・インターフェース・コマンドに関しては次のとおり：

＜ＲＤ１＞－プロセッサへ読込む。

＜ＷＲ１＞－プロセッサから書込む。

＜ＲＡＲ１＞－再書込み後読取りの原子動作。

・更に第１レベルのバス（図５）に関しては次のとおり：

＜ＲＤ２＞－第２レベルのバスからキャッシュ・ラインを要求する。

＜ＲＴＷ２＞－専有所有権を確保した上でキャッシュ・ラインを読取る。

＜ＩＮＶ２＞－第１レベルの全てのプロセッサに対し、キャッシュ・ラインの無効化を命じる。

＜ＷＲ２＞－第２レベルのバス上へキャッシュ・ラインのデータを書出す。

・更に第２レベルのバス（図６）に関しては次のとおり：

＜ＲＤ２＞－第２レベルのバスからキャッシュ・ラインを要求する。

＜ＲＴＷ２＞－専有所有権を確保した上で他のＣＰＵからキャッシュ・ラインを読込む。

＜ＩＮＶ２＞－ローカル・バスである第２レベルのバス上の全てのキャッシュの中のキャッシュ・ラインを無効化する。

＜ＷＲ２＞－プライマリ・キャッシュからキャッシュ・ラインを除去し、或いは、移転する。

＜ＲＤ３＞－第３レベルのバスからキャッシュ・ラインを要求する。

＜ＲＴＷ３＞－専有所有権を確保した上で他のノードからキャッシュ・ラインを読込む。

＜ＩＮＶ３＞－全てのＥＣＵにキャッシュ・ラインの無効化を命じる。

＜ＷＲ３＞－第３レベルのバス上へキャッシュ・ラインを書出す。

【００２８】当業者であれば容易に理解することであるが、本発明は階層化接続の階層数が何層であっても容易に適用し得るものであり、即ち、第４レベル、第５レベル、等々が存在している場合でも適用可能である。

【図面の簡単な説明】

【図１】従来例に係るクラスタ・アーキテクチャを示したハイレベルのブロック図である。

【図２】本発明に係るクラスタ・アーキテクチャを示したハイレベルのブロック図である。

【図３】本発明に係るクラスタ・アーキテクチャの外部一貫性維持装置（ＥＣＵ）の構成要素の中間レベルのブロック図を特に示したより詳細なブロック図である。

【図４】一例として取り上げた具体的なコンピュータ・ファミリーにおけるプライマリ・キャッシュの状態遷移図である。

【図５】図４と同一の具体的なコンピュータ・ファミリーに採用し得るようにした外部一貫性維持装置（ＥＣＵ）の状態遷移図である。

【図６】外部一貫性維持装置の連想記憶のエントリの具体例を示した図である。

【符号の説明】

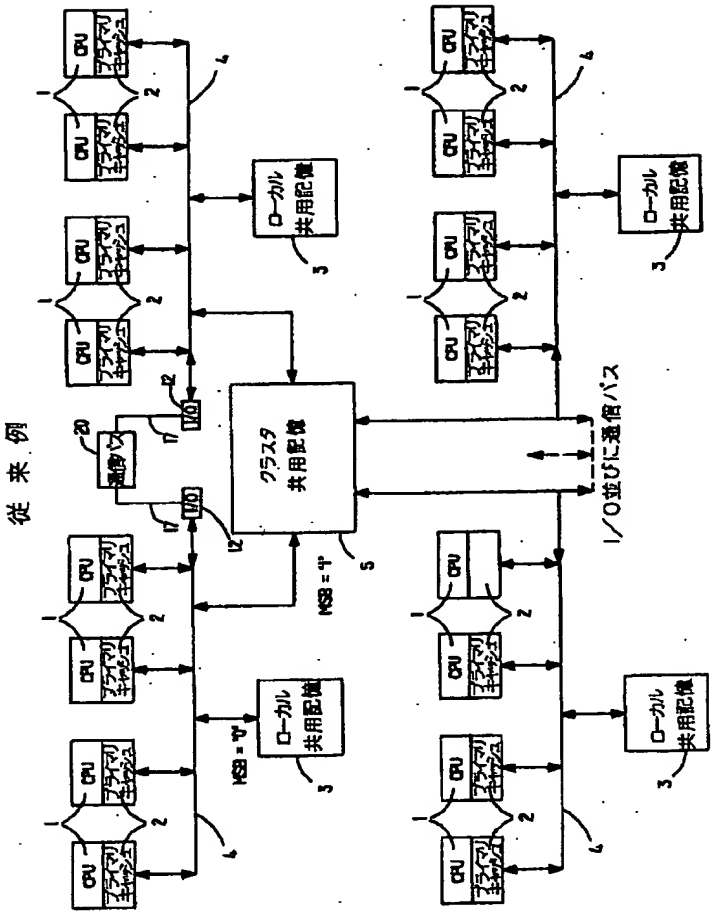
１　中央処理装置（ＣＰＵ）
２　プライマリ・キャッシュ
３　ローカル共用記憶
４　ノード・ローカル・バス
１０　外部一貫性維持装置（ＥＣＵ）
１１　ノード間ＥＣＵバス

【図６】

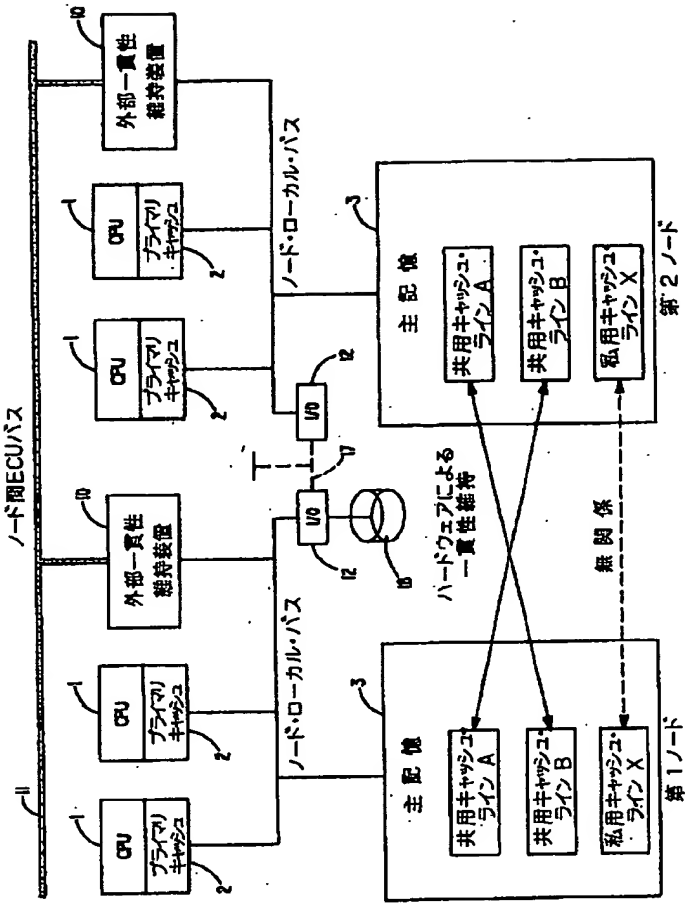
外部一貫性維持装置エントリ（ECUエントリ）
（各共用ページごとに１つずつ）

ノード内の物理ページ・タグ 一義的識別子	ページ内の各キャッシュ・ラインの一貫性状態

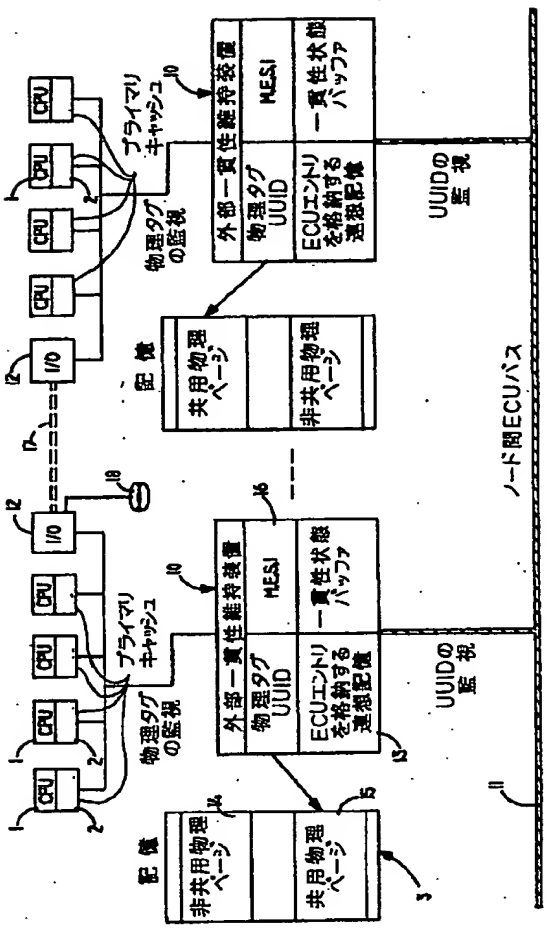
【図1】


従来例
CPU
プライベートキャッシュ
ローカル共用記憶
クラスタ共用記憶
I/O並びに通信バス
MSB = "1"
MSB = "0"

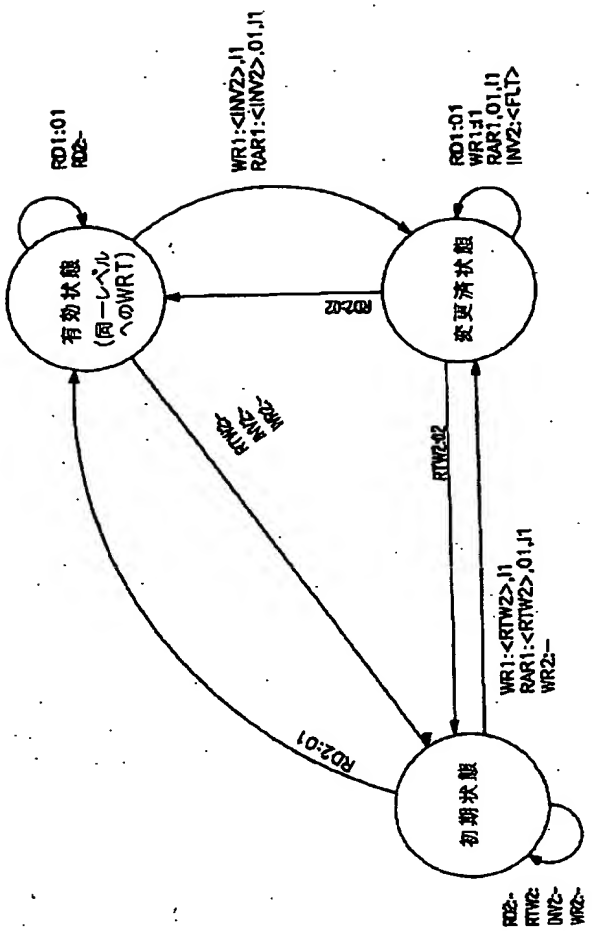
【図２】


ノード間ECUバス
外部一貫性維持装置
CPU
プライマリキャッシュ
ノード・ローカル・バス
I/O
主記憶
共用キャッシュ・ライン A
共用キャッシュ・ライン B
私用キャッシュ・ライン X
第２ノード
第１ノード
ハードウェアによる一貫性維持
関係

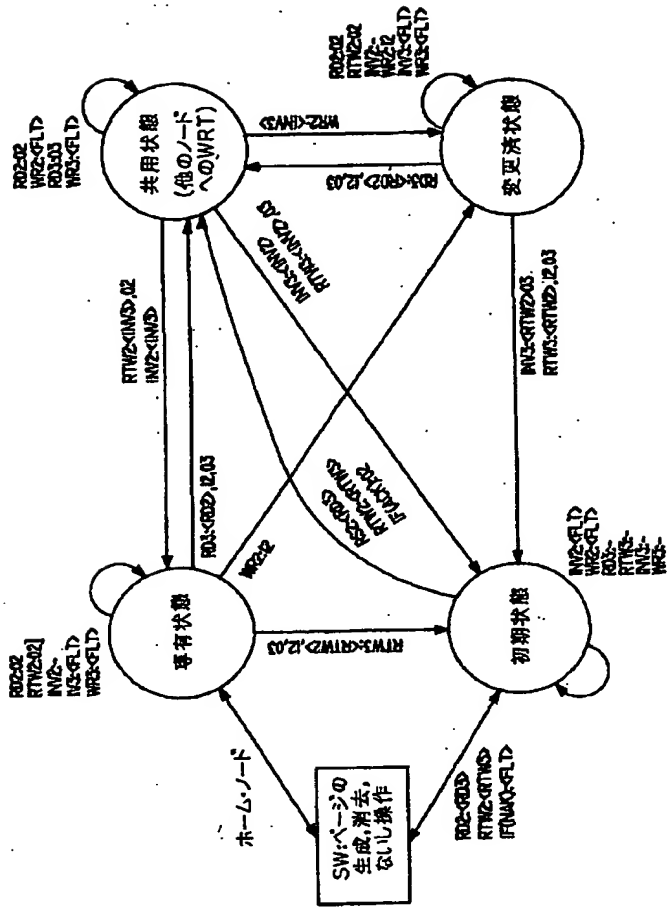
【図3】


CPU
プライマリキャッシュ
物理タグの監視
外部一貫性維持装置
物理タグ UUID
MESI
一貫性状態バッファ
ECUエントリを格納する通知記憶
UUIDの監視
I/O
記憶
共用物理ページ
非共用物理ページ
ノード間ECUバス

【図4】


RD1:01
RD2-
WR1:<INV2>,J1
RAR1:<INV2>,01,J1
RD1:01
WR1:J1
RAR1,01,J1
INV2:<FLT>
有効状態
(同一レベルへのWRT)
変更済状態
RD2:01
RTW2
WR2-
RTW2:02
WR1:<RTW2>,J1
RAR1:<RTW2>,01,J1
WR2=
RD2:01
初期状態
RD2-
RTW2-
INV2-
WR2-

【図5】



フロントページの続き

(72)発明者 ジョン・エイ・ワーツ
アメリカ合衆国アリゾナ州85023，フェニックス，ノース・トゥエンティース・ドライブ 17402